Quickstart: Create a function in Azure with Python using Visual Studio Code

Article • 09/10/2024 • 12 contributors

Feedback

In this article

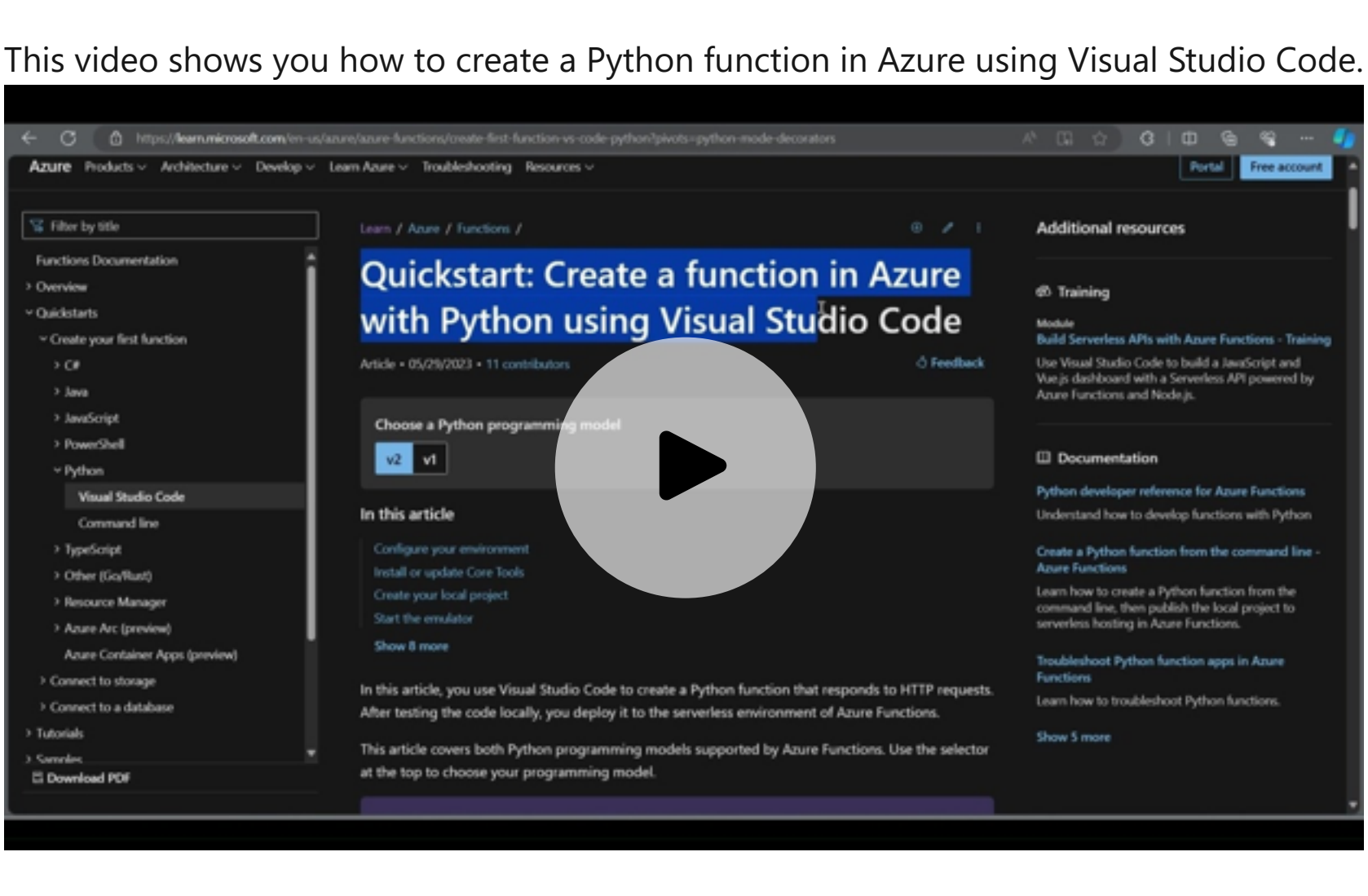
In this article, you use Visual Studio Code to create a Python function that responds to HTTP requests. After testing the code locally, you deploy it to the serverless environment of Azure Functions.

This article uses the Python v2 programming model for Azure Functions, which provides a decorator-based approach for creating functions. To learn more about the Python v2 programming model, see the Developer Reference Guide

Completing this quickstart incurs a small cost of a few USD cents or less in your Azure account.

There's also a CLI-based version of this article.

This video shows you how to create a Python function in Azure using Visual Studio Code.



The steps in the video are also described in the following sections.

Configure your environment

Before you begin, make sure that you have the following requirements in place:

- An Azure account with an active subscription. Create an account for free.
- A Python version that is supported by Azure Functions. For more information, see How to install Python.
- Visual Studio Code on one of the supported platforms.
- The Python extension for Visual Studio Code.
- The Azure Functions extension for Visual Studio Code, version 1.8.1 or later.
- The Azurite V3 extension local storage emulator. While you can also use an actual Azure storage account, this article assumes you're using the Azurite emulator.

Install or update Core Tools

The Azure Functions extension for Visual Studio Code integrates with Azure Functions Core Tools so that you can run and debug your functions locally in Visual Studio Code using the Azure Functions runtime. Before getting started, it's a good idea to install Core Tools locally or update an existing installation to use the latest version.

In Visual Studio Code, select F1 to open the command palette, and then search for and run the command **Azure Functions: Install or Update Core Tools**.

This command tries to either start a package-based installation of the latest version of Core Tools or update an existing package-based installation. If you don't have npm or Homebrew installed on your local computer, you must instead manually install or update Core Tools.

Create your local project

In this section, you use Visual Studio Code to create a local Azure Functions project in Python. Later in this article, you publish your function code to Azure.

1. In Visual Studio Code, press `F1` to open the command palette and search for and run the command `Azure Functions: Create New Project...`.
2. Choose the directory location for your project workspace and choose **Select**. You should either create a new folder or choose an empty folder for the project workspace. Don't choose a project folder that is already part of a workspace.
3. Provide the following information at the prompts:

Expand table

Prompt	Selection
Select a language	Choose `Python (Programming Model V2)`.
Select a Python interpreter to create a virtual environment	Choose your preferred Python interpreter. If an option isn't shown, type in the full path to your Python binary.
Select a template for your project's first function	Choose `HTTP trigger`.
Name of the function you want to create	Enter `HttpExample`.
Authorization level	Choose `ANONYMOUS`, which lets anyone call your function endpoint. For more information, see Authorization level.
Select how you would like to open your project	Choose `Open in current window`.

4. Visual Studio Code uses the provided information and generates an Azure Functions project with an HTTP trigger. You can view the local project files in the Explorer. The generated `function_app.py` project file contains your functions.
5. In the local.settings.json file, update the `AzureWebJobsStorage` setting as in the following example:

JSON

```json
"AzureWebJobsStorage": "UseDevelopmentStorage=true",
```

This tells the local Functions host to use the storage emulator for the storage connection required by the Python v2 model. When you publish your project to Azure, this setting uses the default storage account instead. If you're using an Azure Storage account during local development, set your storage account connection string here.

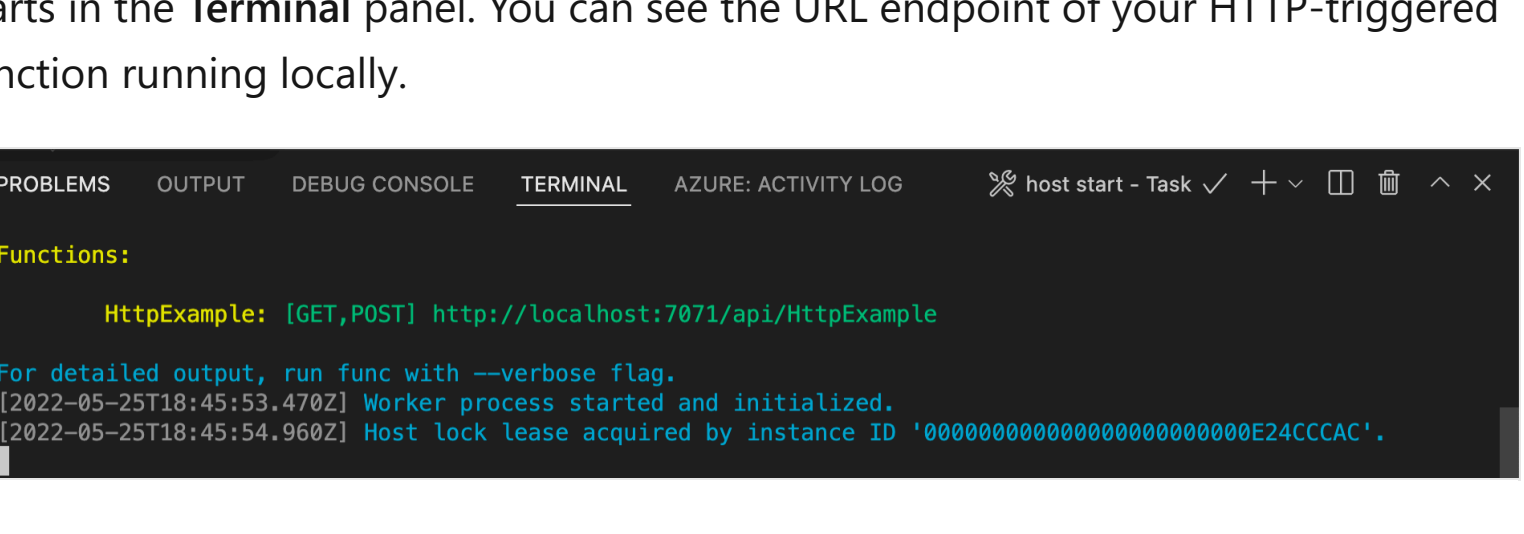
Start the emulator

1. In Visual Studio Code, press `F1` to open the command palette. In the command palette, search for and select `Azurite: Start`.
2. Check the bottom bar and verify that Azurite emulation services are running. If so, you can now run your function locally.

Run the function locally

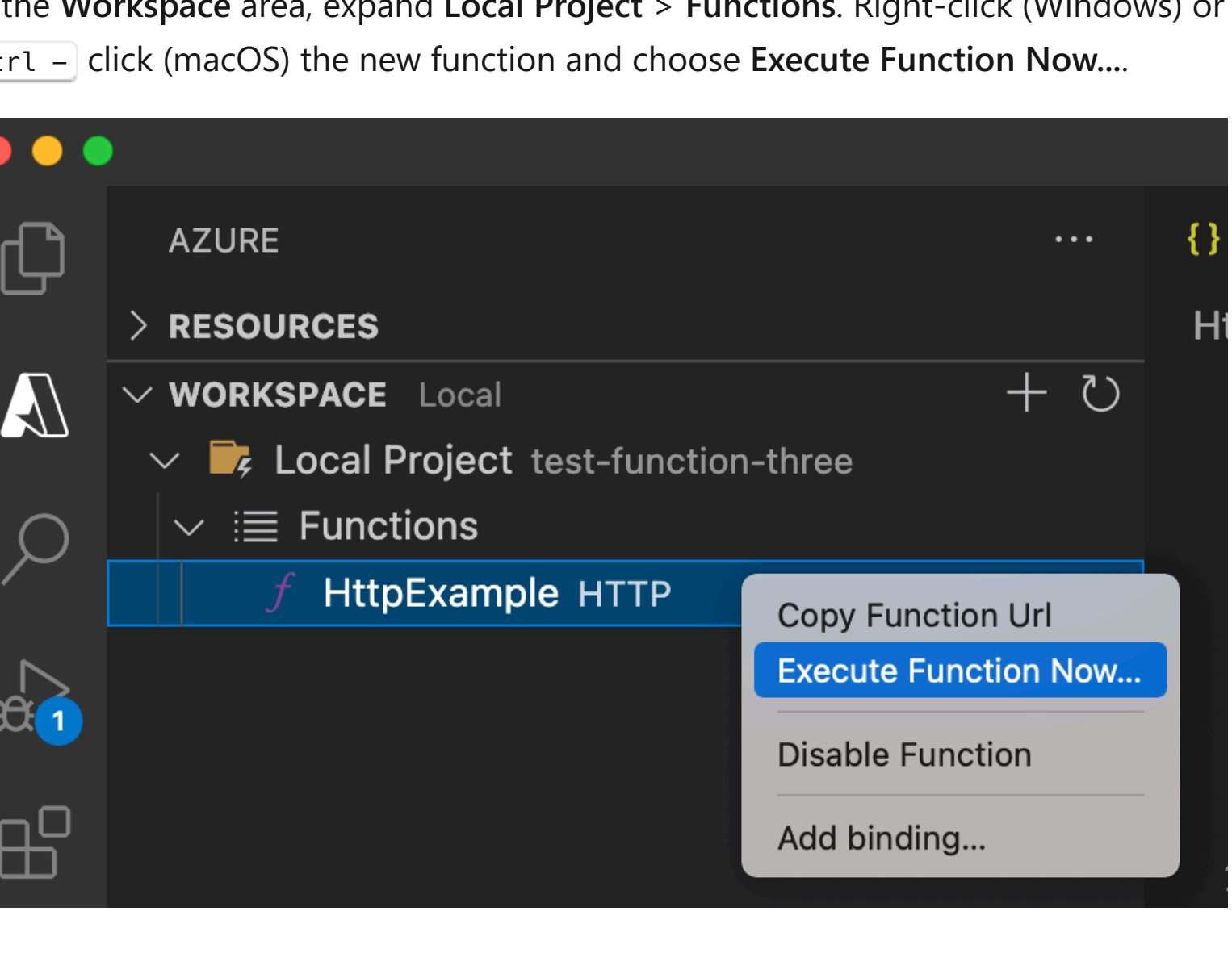
Visual Studio Code integrates with Azure Functions Core tools to let you run this project on your local development computer before you publish to Azure.

1. To start the function locally, press `F5` or the **Run and Debug** icon in the left-hand side Activity bar. The **Terminal** panel displays the Output from Core Tools. Your app starts in the **Terminal** panel. You can see the URL endpoint of your HTTP-triggered function running locally.



If you have trouble running on Windows, make sure that the default terminal for Visual Studio Code isn't set to **WSL Bash**.

2. With Core Tools still running in **Terminal**, choose the Azure icon in the activity bar. In the **Workspace** area, expand **Local Project** > **Functions**. Right-click (Windows) or `Ctrl -` click (macOS) the new function and choose **Execute Function Now...**.



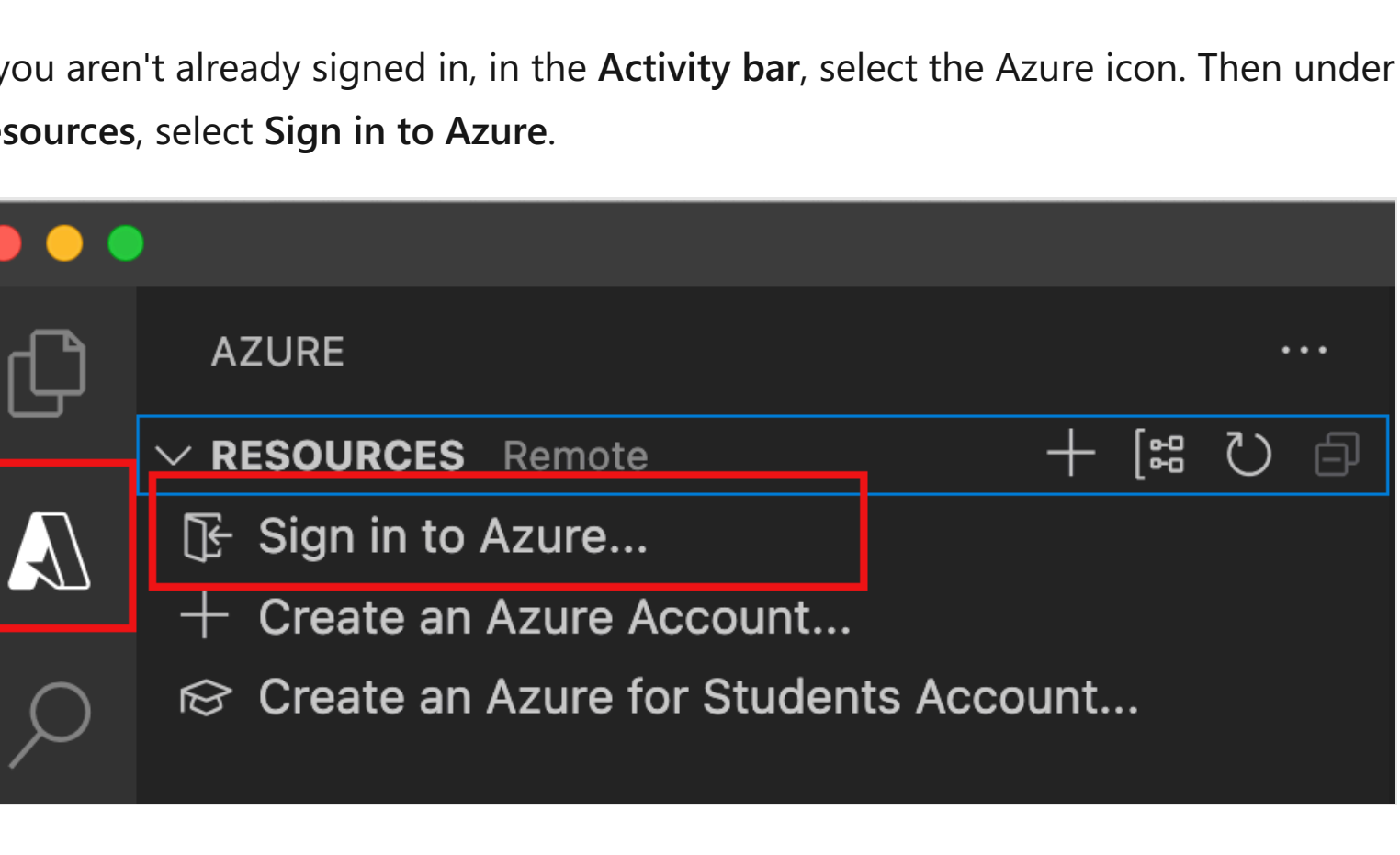
3. In **Enter request body** you see the request message body value of `{ "name": "Azure" }`. Press Enter to send this request message to your function.
4. When the function executes locally and returns a response, a notification is raised in Visual Studio Code. Information about the function execution is shown in **Terminal** panel.
5. With the **Terminal** panel focused, press `Ctrl + C` to stop Core Tools and disconnect the debugger.

After you verify that the function runs correctly on your local computer, it's time to use Visual Studio Code to publish the project directly to Azure.

Sign in to Azure

Before you can create Azure resources or publish your app, you must sign in to Azure.

1. If you aren't already signed in, in the **Activity bar**, select the Azure icon. Then under **Resources**, select **Sign in to Azure**.



If you're already signed in and can see your existing subscriptions, go to the next section. If you don't yet have an Azure account, select **Create an Azure Account**. Students can select **Create an Azure for Students Account**.

2. When you are prompted in the browser, select your Azure account and sign in by using your Azure account credentials. If you create a new account, you can sign in after your account is created.
3. After you successfully sign in, you can close the new browser window. The subscriptions that belong to your Azure account are displayed in the side bar.

Create the function app in Azure

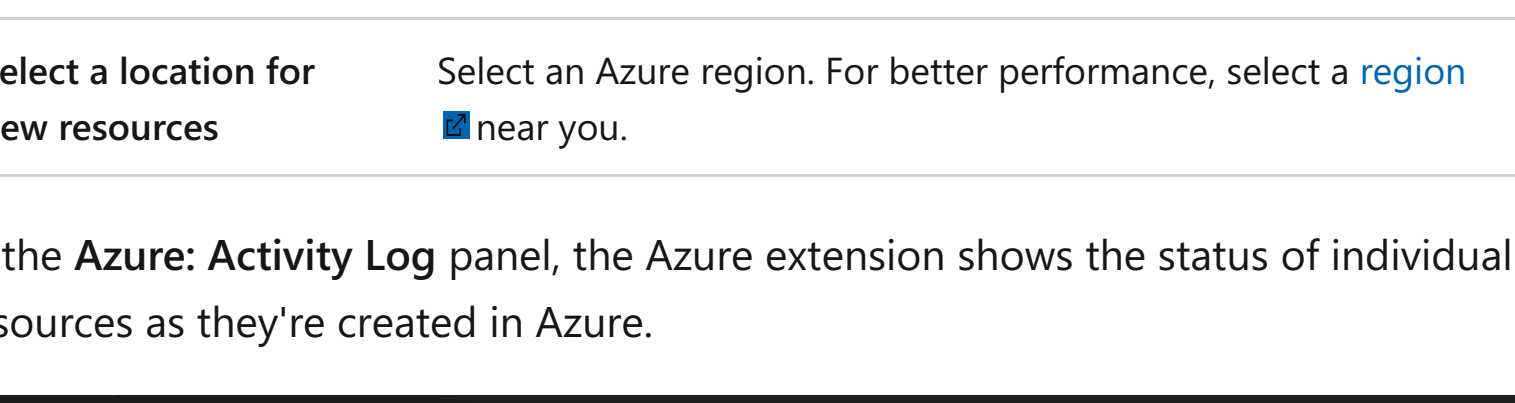
In this section, you create a function app and related resources in your Azure subscription. Many of the resource creation decisions are made for you based on default behaviors. For more control over the created resources, you must instead create your function app with advanced options.

1. In Visual Studio Code, select F1 to open the command palette. At the prompt (`>`), enter and then select **Azure Functions: Create Function App in Azure**.
2. At the prompts, provide the following information:

Expand table

Prompt	Action
Select subscription	Select the Azure subscription to use. The prompt doesn't appear when you have only one subscription visible under **Resources**.
Enter a globally unique name for the function app	Enter a name that is valid in a URL path. The name you enter is validated to make sure that it's unique in Azure Functions.
Select a runtime stack	Select the language version you currently run locally.
Select a location for new resources	Select an Azure region. For better performance, select a region near you.

In the **Azure: Activity Log** panel, the Azure extension shows the status of individual resources as they're created in Azure.



3. When the function app is created, the following related resources are created in your Azure subscription. The resources are named based on the name you entered for your function app.
   - A resource group, which is a logical container for related resources.
   - A standard Azure Storage account, which maintains state and other information about your projects.
   - A function app, which provides the environment for executing your function code. A function app lets you group functions as a logical unit for easier management, deployment, and sharing of resources within the same hosting plan.
   - An Azure App Service plan, which defines the underlying host for your function app.
   - An Application Insights instance that's connected to the function app, and which tracks the use of your functions in the app.

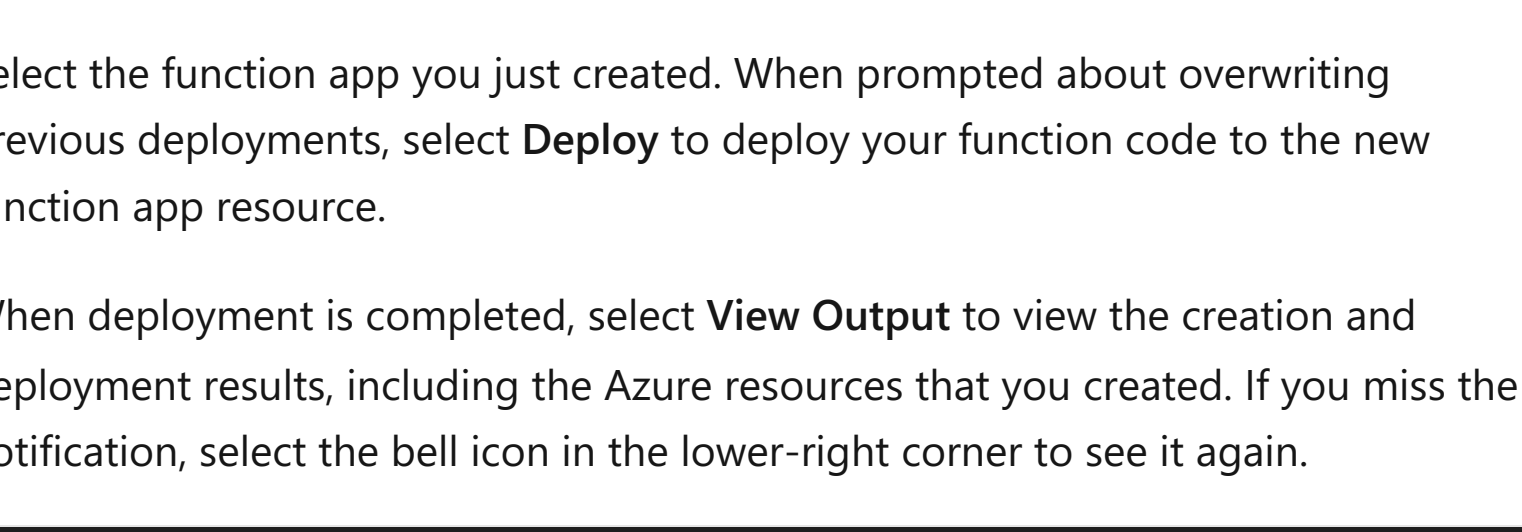
A notification is displayed after your function app is created and the deployment package is applied.

> **Tip**
>
> By default, the Azure resources required by your function app are created based on the name you enter for your function app. By default, the resources are created with the function app in the same, new resource group. If you want to customize the names of the associated resources or reuse existing resources, publish the project with advanced create options.

Deploy the project to Azure

> **Important**
>
> Deploying to an existing function app always overwrites the contents of that app in Azure.

1. In the command palette, enter and then select **Azure Functions: Deploy to Function App**.
2. Select the function app you just created. When prompted about overwriting previous deployments, select **Deploy** to deploy your function code to the new function app resource.
3. When deployment is completed, select **View Output** to view the creation and deployment results, including the Azure resources that you created. If you miss the notification, select the bell icon in the lower-right corner to see it again.



Run the function in Azure

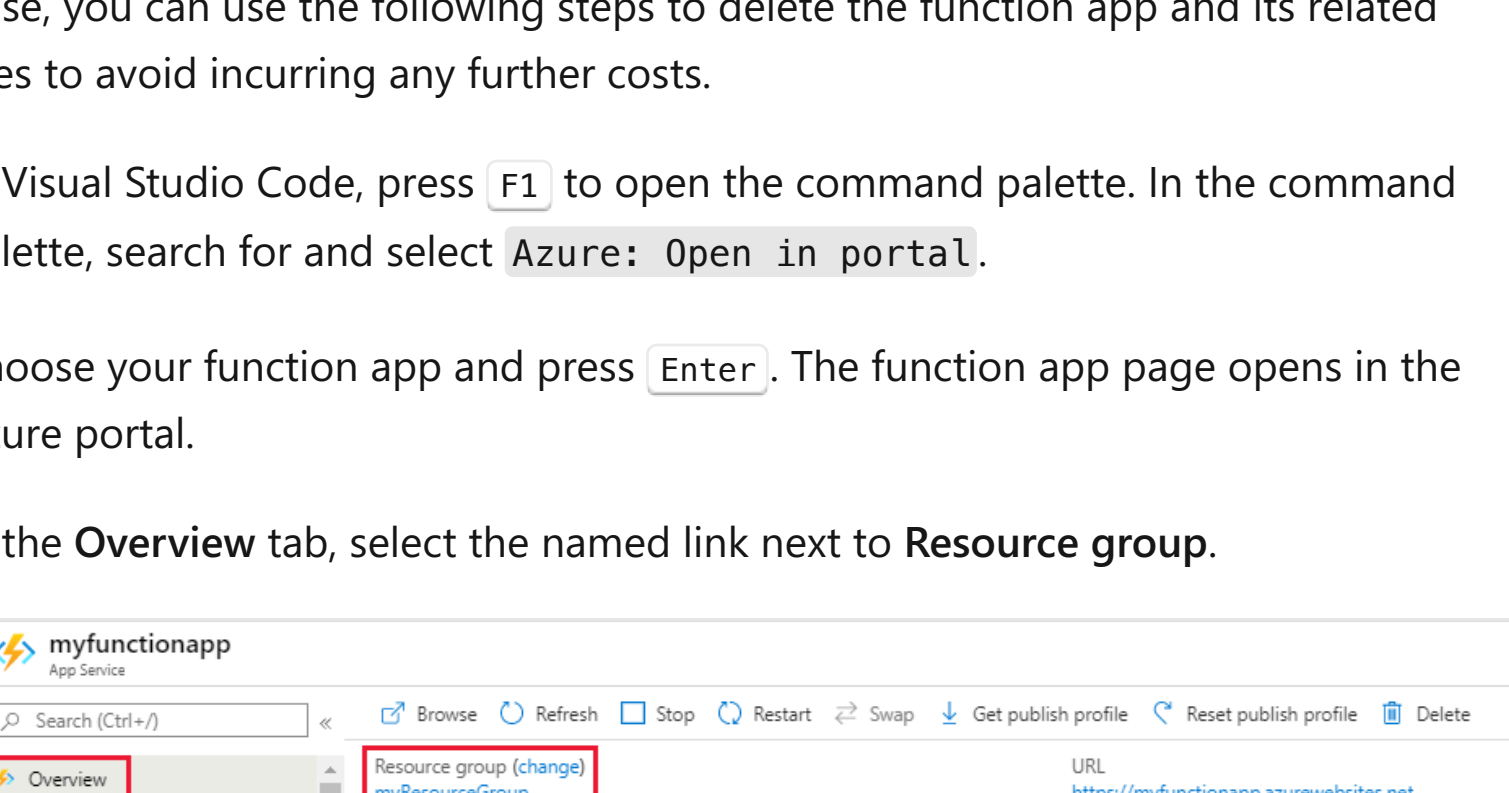
1. Press `F1` to display the command palette, then search for and run the command `Azure Functions:Execute Function Now...`. If prompted, select your subscription.
2. Select your new function app resource and `HttpExample` as your function.
3. In **Enter request body** type `{ "name": "Azure" }`, then press Enter to send this request message to your function.
4. When the function executes in Azure, the response is displayed in the notification area. Expand the notification to review the full response.

Clean up resources

When you continue to the next step and add an Azure Storage queue binding to your function, you'll need to keep all your resources in place to build on what you've already done.

Otherwise, you can use the following steps to delete the function app and its related resources to avoid incurring any further costs.

1. In Visual Studio Code, press `F1` to open the command palette. In the command palette, search for and select `Azure: Open in portal`.
2. Choose your function app and press `Enter`. The function app page opens in the Azure portal.
3. In the **Overview** tab, select the named link next to **Resource group**.



4. On the **Resource group** page, review the list of included resources, and verify that they're the ones you want to delete.
5. Select **Delete resource group**, and follow the instructions.

   Deletion may take a couple of minutes. When it's done, a notification appears for a few seconds. You can also select the bell icon at the top of the page to view the notification.

For more information about Functions costs, see Estimating Consumption plan costs.

Next steps

You created and deployed a function app with a simple HTTP-triggered function. In the next articles, you expand that function by connecting to a storage service in Azure. To learn more about connecting to other Azure services, see Add bindings to an existing function in Azure Functions.

Connect to Azure Cosmos DB

Connect to an Azure Storage queue

Having issues? Let us know.

Note: The author created this article with assistance from AI. Learn more

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Additional resources

Documentation

Create a Python function from the command line - Azure Functions

Learn how to create a Python function from the command line, then publish the local project to serverless hosting in Azure Functions.

Python developer reference for Azure Functions

Understand how to develop, validate, and deploy your Python code projects to Azure Functions using the Python library for Azure Functions.

Develop Azure Functions by using Visual Studio Code

Learn how to develop and test Azure Functions by using the Azure Functions extension for Visual Studio Code.

Show 4 more

Training

Learning path
Create serverless applications learning path - Training

In this learning path, discover Azure Functions that create event-driven, compute-on-demand systems using server-side logic to build serverless architectures.

Certification
Microsoft Certified: Azure Developer Associate - Certifications

Build end-to-end solutions in Microsoft Azure to create Azure Functions, implement and manage web apps, develop solutions utilizing Azure storage, and more.